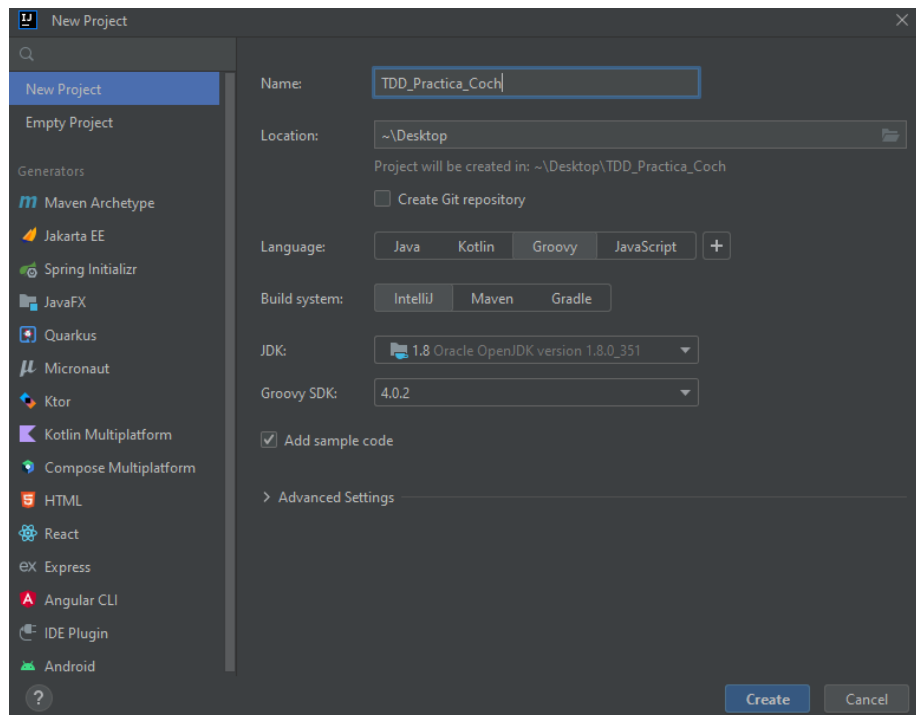


Memoria

Mi Primer TDD UND 10

Creación del proyecto, usare Groovy.



Importo JUNIT5 para poder utilizar los Test y las Assertions

```
import org.junit.jupiter.api.Assertions;  
import org.junit.jupiter.api.Test;
```

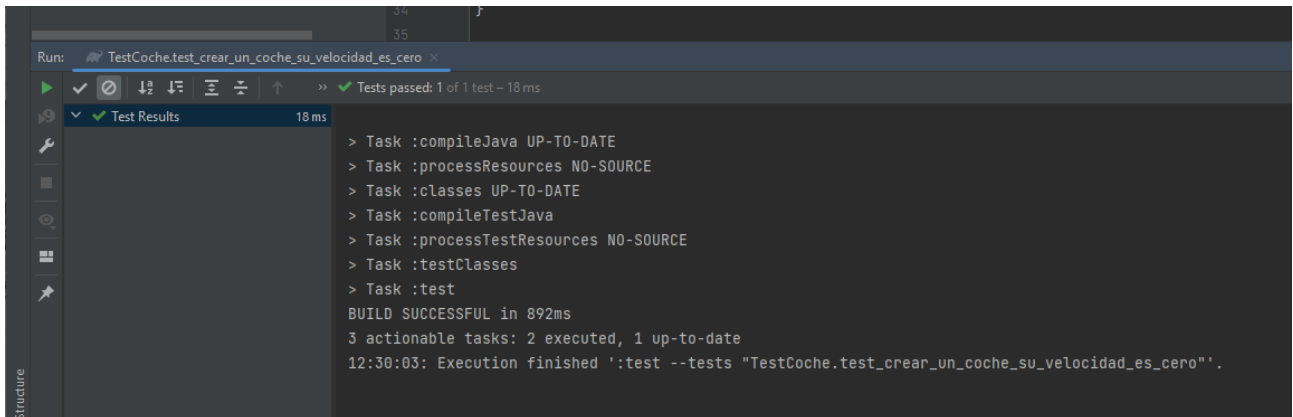
Creo el primer test, consiste en que al crear un nuevo objeto Coche su velocidad inicial debe ser 0.

```
public class TestCoche {  
    @Test  
    public void test_crear_un_coche_su_velocidad_es_cero(){  
        Coche nuevoCoche = new Coche();  
        Assertions.assertEquals( expected: 0, nuevoCoche.velocidad);  
    }  
}
```

Y esta es la clase, al no darle valor a velocidad la coge por 'null' es decir nada, 0.

```
2 usages  Nicolas  
public class Coche {  
    5 usages  
    public int velocidad;
```

Ha pasado el Test.



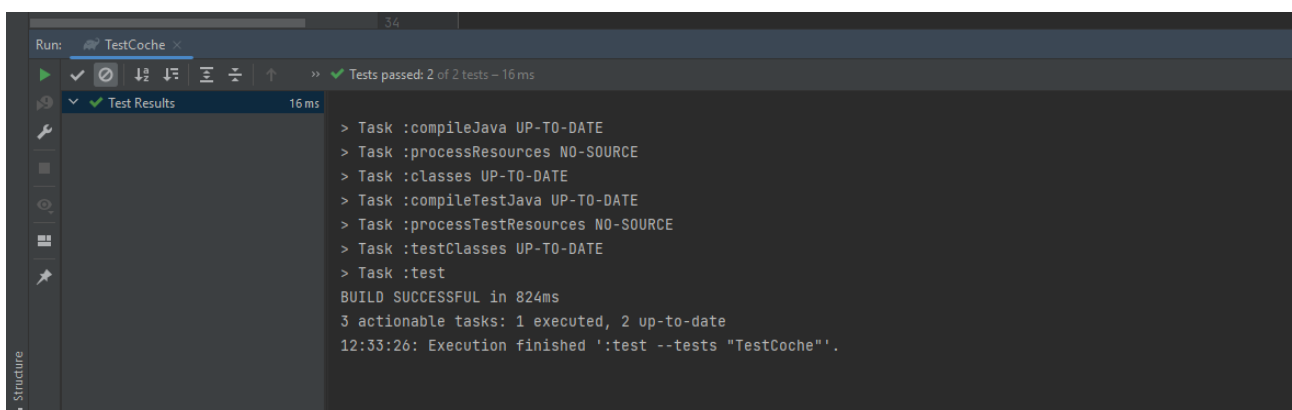
Ahora el siguiente Test es que el coche aumente su velocidad. Para hacerlo se le da un valor de aceleración y comprueba que la velocidad del coche es igual a la velocidad acelerada.

```
@Test
public void test_al_acelerar_un_coche_su_velocidad_aumenta(){
    Coche nuevoCoche = new Coche();
    nuevoCoche.acelerar( aceleracion: 30);
    Assertions.assertEquals( expected: 30, nuevoCoche.velocidad);
}
```

El método usado para acelerar la velocidad del coche.

```
1 usage  Nicolas
public void acelerar(int aceleracion) {
    velocidad += aceleracion;
}
```

Ha pasado el test.



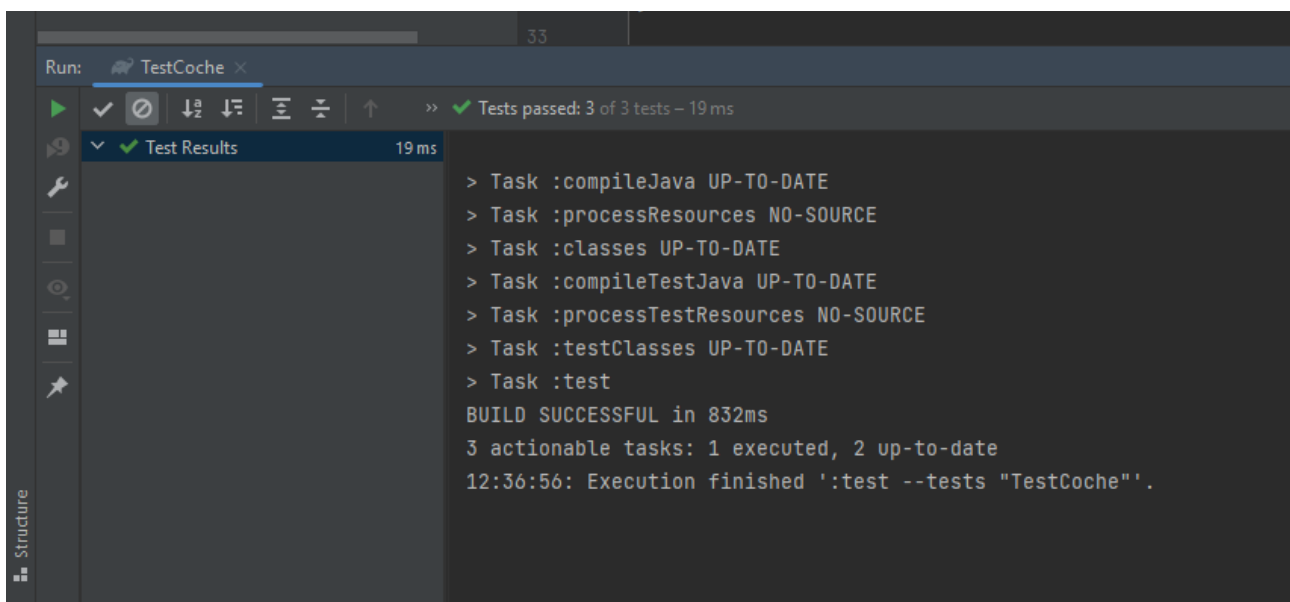
Lo siguiente que vamos a comprobar es que al frenar la velocidad del coche disminuya. Para eso se le añade un valor de desaceleración, le aumentamos la velocidad inicial a 50 y le restamos 20, el resultado esperado es 30.

```
@Test
public void test_al_decelerar_un_coche_su_velocidad_disminuye(){
    Coche nuevoCoche = new Coche();
    nuevoCoche.velocidad = 50;
    nuevoCoche.decelerar( deceleracion: 20);
    Assertions.assertEquals( expected: 30, nuevoCoche.velocidad);
}
```

El método usado para la desaceleración.

```
public void decelerar(int deceleracion) {
    velocidad -= deceleracion;
}
```

Ha pasado el Test.



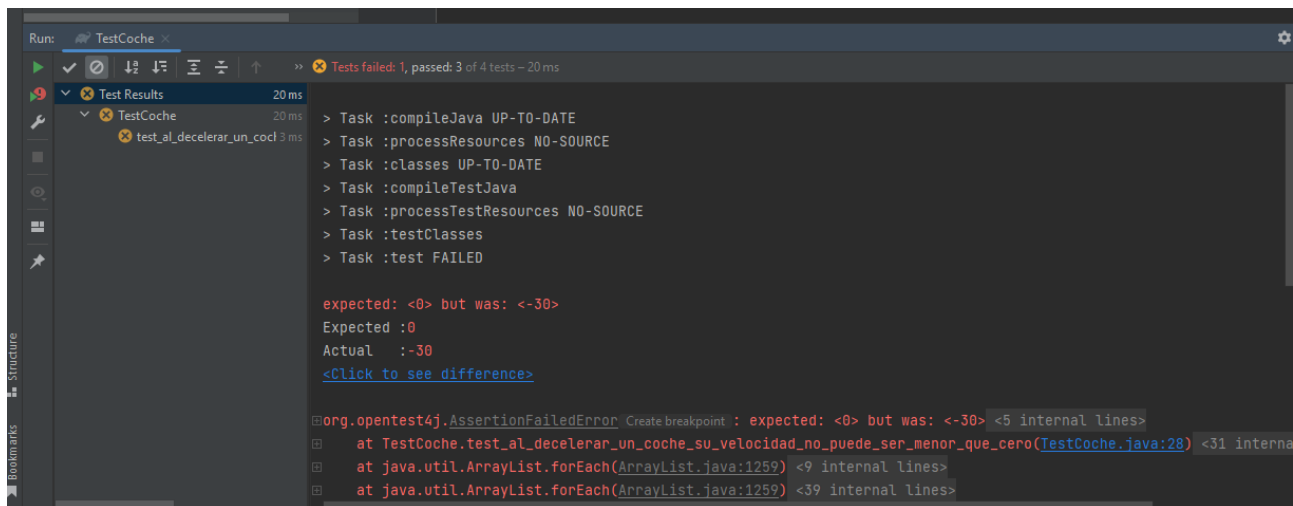
Y el ultimo Test sera comprobar que la velocidad del coche no puede ser negativa. Le daremos un valor inicial de 50 y le restaremos 80, el valor esperado debería ser 0 ya que no queremos que sea inferior.

```

Nicolas
@Test
public void test_al_decelerar_un_coche_su_velocidad_no_puede_ser_menor_que_cero(){
    Coche nuevoCoche = new Coche();
    nuevoCoche.velocidad = 50;
    nuevoCoche.decelerar( deceleracion: 80);
    Assertions.assertEquals( expected: 0, nuevoCoche.velocidad);
}

```

No ha pasado el Test ya que el resultado es -30, vamos a arreglarlo en el método de desaceleración.



```

Run: TestCoche
Tests failed: 1, passed: 3 of 4 tests - 20 ms
Test Results
  TestCoche
    test_al_decelerar_un_cocl 3 ms
    > Task :compileJava UP-TO-DATE
    > Task :processResources NO-SOURCE
    > Task :classes UP-TO-DATE
    > Task :compileTestJava
    > Task :processTestResources NO-SOURCE
    > Task :testClasses
    > Task :test FAILED
    expected: <0> but was: <-30>
    Expected :0
    Actual   :-30
    <Click to see difference>
    org.opentest4j.AssertionFailedError: Create breakpoint : expected: <0> but was: <-30> <5 internal lines>
    at TestCoche.test_al_decelerar_un_coche_su_velocidad_no_puede_ser_menor_que_cero(TestCoche.java:28) <31 internal lines>
    at java.util.ArrayList.forEach(ArrayList.java:1259) <9 internal lines>
    at java.util.ArrayList.forEach(ArrayList.java:1259) <39 internal lines>

```

Añadiéndole un If que compruebe la velocidad, cuando sea inferior a 0 la convertirá a 0

```

2 usages Nicolas
public void decelerar(int deceleracion) {
    velocidad -= deceleracion;
    if (velocidad < 0) velocidad = 0;
}

```

Ahora si ha pasado la prueba.

